(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 March 2001 (08.03.2001)

PCT

(10) International Publication Number
WO 01/16874 A1

(51) International Patent Classification⁷: G06K 19/06, 05/00, 07/00

(21) International Application Number: PCT/US00/00084

(22) International Filing Date: 5 January 2000 (05.01.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/386,287 31 August 1999 (31.08.1999) US

(71) Applicant: CRYPTEC SYSTEMS, INC. [US/US]; 475 Alberto Way, Los Gatos, CA 95032 (US).

(72) Inventor: CARPER, Todd; 19834 Merritt Drive, Cupertino, CA 95014 (US).

(74) Agent: WHITT, Stephen, R.; 1215 Tottenham Court, Reston, VA 20194 (US).

(81) Designated State *(national)*: SG.

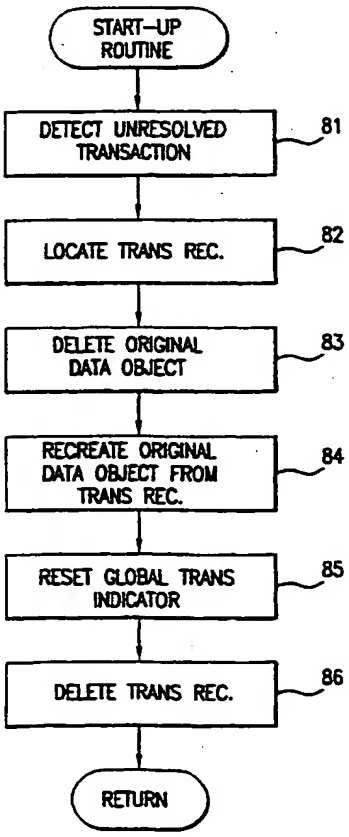
(84) Designated States *(regional)*: European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published:
— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SMART CARD TRANSACTION MANAGER

(57) Abstract: A transaction manager (61) is provided in a smart card operating system which ensures data integrity during smart card transactions which change the value of a data object (41) stored on the smart card (10).



WO 01/16874 A1

SMART CARD TRANSACTION MANAGER

FIELD OF THE INVENTION

The present invention relates to the field of portable tokens, such as smart cards. More particularly, the present invention relates to a transaction manager preferably found in a smart card operating system which assures the integrity of data stored on the card.

BACKGROUND OF THE INVENTION

Most consumers are now familiar with stored value cards. Stored value cards are purchased from a vendor, or issued from a vending machine with a specific monetary value. When the card user desires to purchase goods or services, the card is presented at the point of sale, and the cost of the goods or services is deducted from the value of the card. The cardholder may continue to use the stored value card in this manner until all the value has been removed from the card. The card may then be discarded, or its value may be replenished. Such cards are commonly used to pay subway fares or make phone calls.

Many stored value cards resembled credit card. They have a plastic body with a magnetic strip on the back. When used in a transaction, the magnetic strip is read from and/or written to by an external device, generically referred to hereafter as a terminal. Consumer familiarity and the ease with which such cards may be carried, makes them ideal for a wide variety of "cash-less" transactions.

Unfortunately, the magnetic strip encoding of data on stored value cards does not provide a high degree of data security. Furthermore, the magnetic strip offers little, if any, real processing capability beyond simply storing data.

Thus, the current trend is away from conventional stored value cards, and into a class of devices generally called smart cards. Rather than employing information encoded on a magnetic strip, smart cards include a microprocessor with a memory element embedded within a plastic card. With a microprocessor, smart cards are capable of interacting with a wide variety terminals to accomplish a broad range of transactions. Compared with conventional stored value cards, smart cards are able to communicate a broader and a more detailed range of information regarding the cardholder, the cardholder account(s), the card issuer, transaction authorization(s) and capabilities, etc.

The term "smart card" is used throughout as a convenient name for a broad class of devices sometimes referred to as portable tokens. Smart cards are a common, present form of portable tokens, but as will be seen hereafter the actual physical form of the portable token, as well as the specific means by which the portable token communicates data to the outside world are not the subject of the present invention.

Fig 1 shows an exemplary smart card 10. Roughly the size of a credit card, smart card 10 includes a built-in microprocessor 12 with an integral memory element, and conductive contacts 13. Microprocessor 12 is typically a single wafer integrated circuit (IC) mounted on, or embedded within the otherwise plastic smart card. Conductive contacts 13 interface with a terminal to electrically transfer data between the terminal and the smart card. Other embodiments of the smart card do not include conductive contacts 13. Such "contactless" smart cards receive information via proximately coupling, such as magnetic coupling, or via remote coupling, such as radio communication.

The microprocessor 12 and conductive contacts 13 of Fig 1, are shown in some additional detail in Fig 2. Conductive contacts variously include power contacts, at least one input/output (I/O) port, a reset port, and a clock (clk) signal port. Microprocessor 12 comprises a central processing unit (CPU) 21 which is generically control logic including I/O circuitry 23. Terminal signals variously interface with CPU 21 through the conductive contacts 13 and I/O circuitry 23. Microprocessor 12 further comprises a memory element 20, typically including Random Access Memory (RAM) 22, Read Only Memory (ROM) 24, and Electrically Erasable Programable Read Only Memory (EEPROM) 26.

Operating power, a user input keypad, and a display for the smart card microprocessor are provided by an external device such as; an ATM, merchant point-of-sale device, an off-card controller, a security control device, etc. Hereafter, all external devices communicating data to/from a smart card, regardless of their size, form factor, or I/O connection type, are referred to as "terminals."

Generally, a terminal must include a mechanism detecting the presence of a properly positioned smart card. Upon detecting the smart card, the terminal provides power to the microprocessor, and typically sends a reset (RST) signal to the smart card. The smart card then resets, and/or initiates an internal initialization function. After reset, the smart card returns an answer-to-reset (ATR) signal to the terminal. The nature and

content of ATR signal has been established by convention. International Standards Organization (ISO) standard 7816 defines the ATR signal which communicates basic information concerning the smart card to the terminal. Once such basic information is successfully recognized by the terminal, data transfer between the smart card and the terminal can be established.

Like the ATR signal, the physical structure of smart cards is established by ISO-7816. This standard specifies the physical characteristics of smart cards such as the size, shape, composition, placement of electrical contacts, the electrical interface, the method of data transmission for smart cards, i.e. T=0 or T=1 data transmission modes, etc. While ISO standard 7816 has largely led to uniformity in the physical construction and communication protocol of smart cards, the standard does not effectively specify an operating system or an application programming format.

Security is an ever-present issue in smart card programming and use. This is particularly true when smart cards are used to conduct financial transactions. To inspire bank, merchant, and cardholder confidence in smart card technology, smart cards must provide security features which prevent unauthorized use or counterfeiting of a card. When used to store value, smart cards must prevent users from fraudulently adding value to a smart card.

When functioning as stored value cards, conventional smart cards can be programmed and re-programmed to contain a particular value as desired by the cardholder. This value is gradually depleted as purchases are made. A merchant terminal at a point of sale may be able to simply deduct value from the smart card, or the smart card may be designed to require the cardholder to input a personal identification number (PIN) before value is deducted from the card. In either event, conventional smart cards used as stored value cards are subject to constant security attacks.

In one class of favored attack, generally referred to a "card yank," a fraudulent user, or "hacker" will monitor the memory state of a smart card during critical periods of a financial transaction in which data is being written to the smart card memory. Within such periods, the memory may, in whole or in part, transition through an undefined state.

Once potential periods of undefined memory state are identified, the hacker terminates the transaction during these periods in an attempt to acquire an undefined, or

wrongly defined memory state. The undefined, or wrongly defined memory state may later be interrupted in a subsequent transaction as having a substantially higher value than that otherwise authorized had the "yanked" transaction run to completion.

There are other attack methodologies which attempt to create, and then to exploit a loss of data integrity within the smart card memory. Most of these methodologies seek to create undefined or wrongly defined data state by interrupting a transaction, or a specific sequence of commands within the transaction.

The problem of smart card data integrity may arise in circumstances where a terminal fails during a transaction. Loss of power, or some other "glitch" within the terminal may result in an undefined memory state for a smart card in use during the terminal failure. Here, assuming the smart card is being used as a stored value card, the undefined memory state may be interpreted in a subsequent transaction as having either a higher value, or a lower value than that properly assignable to the card. Either situation is unacceptable.

An example illustrating the problem of conventional smart card data integrity will now be presented with reference to Figs. 3A and 3B. In these examples, it is assumed that the smart card is being used as a stored value card.

When used as a stored value card, conventional smart cards typically used one of two approaches. In a first approach illustrated in Fig. 3A, at least one block of EEPROM 30 is rewritten every time a new value is stored on the card. The EEPROM used in conventional smart cards has a minimum data block size of 16 bytes. Thus, at least 16 bytes of EEPROM must be rewritten every time a new value is stored on the smart card. This may take up to 10 milliseconds.

The 16 byte EEPROM data block 30 of Fig. 3A is nominally implemented in a number of memory cells (4 cells are indicated in Fig 3A). Every value capable of being stored in this EEPROM data block will produce a unique array of data bits across the combined set of memory cells. Since each bit-storing element in the memory cell is non-volatile, i.e., it remains persistent until changed, an opportunity exists to interrupt the process of rewriting the entire array of data bits, and thereby create an erroneous stored value.

Looking at Fig 3A, the raster 31 traversing the array of memory cells forming the

EEPROM data block conceptually illustrates the rewriting process. By monitoring known voltage levels in the EEPROM, a hacker is able to determine when the rewriting process is underway. Thus, if power is terminated, i.e., the smart card is yanked, at some point during the rewriting process, shown as point 32 in Fig 3A, the data actually stored in the memory cell array will be a combination of the partially rewritten data and partially remaining "old," i.e., not-rewritten data. This combination of data may be subsequently interrupted as having a value very different from the data originally stored on the smart card, or the new data which should have been stored on the smart card had the transaction run to completion. If the original value is very low, the hacker has potentially much to gain and very little to lose by repeatedly yanking a smart card.

In a second approach to implementing a stored value function, conventional smart cards will alternate storing the value between two data blocks "A" and "B" of EEPROM 34 in Fig 3B. Each of A and B has the 16 byte structure explained above. The smart card also includes a flag stored elsewhere in EEPROM indicating which one of data blocks A and B currently contains valid stored data. This indicator flag is also rewritten every time a new value is written to the smart card. Thus, is addition to the attack explained above with respect to Fig. 3A, the approach illustrated in Fig. 3B is subject to attack during the process of rewriting the indicator flag.

The stored value, financial transaction context used above is but one convenient example illustrating the problem of smart card data integrity. As smart cards continue to evolve, they will be required to run multiple, disparate applications from different sources. Such applications must be run is such a manner that ensures data integrity not only from outside attacks, but also from data conflicts between the different applications.

SUMMARY OF THE INVENTION

The present invention ensures data integrity for smart card data objects changed during a transaction, or changed by operation of a smart card application or the OS.

In one aspect, the present invention provides a method of securely performing a smart card transaction upon receiving a command from a terminal, where the command requires change of an original data object stored in smart card memory. The method comprises copying the original data object into a transaction record, and setting a transaction record indicator. The method may also comprise, after setting the transaction record indicator, changing the original data object, and, after successfully changing the original data object, deleting the transaction record and resetting the transaction record indicator.

This method is well adapted for use in smart card's using EEPROM memory. The transaction record indicator contemplated above may be a global data object established in a reserved portion of EEPROM.

Within the method of the present invention, the original data object may be a data record stored in a file directory established in EEPROM. Thus, when copying the original data object into the transaction record the method creates a transaction data record in the file directory, where the transaction data record may identify an address in memory at which a copy of the original data record is located.

Alternately, the original data object may comprise an original data record stored in a file directory in memory, and additional data associated with the original data record. Thus, when copying the original data object into the transaction record, the method creates a transaction data record in the file directory, wherein the transaction data record contains a field identifying at least one address in memory at which a copy of the original data record and a copy of the additional associated data are located.

In another aspect, the present invention provides a method of executing a transaction between a terminal and a smart card, the smart card comprising a microprocessor and a memory, the memory storing an application and an operating system (OS), the OS comprising a transaction manager. The method comprises receiving a command in the OS from the terminal, wherein the command requires changing an original data object, communicating the command from the OS to the application, calling

the transaction manager from the application, by operation of the transaction manager, copying the original data object into a transaction record, and setting a transaction record indicator in memory.

The method may also include, after setting the transaction record indicator, returning to the application and changing the original data object in accordance with the command, after successfully changing the original data object, calling the transaction manager, and by operation of the transaction manager, deleting the transaction record and resetting the transaction record indicator.

The command may comprise a series of commands received in the OS from the terminal and communicated to the application, wherein during the series of commands the application calls the transaction manager, such that the OS functions as a true OS, being incapable of executing any one of the series of commands received from the terminal.

In another aspect, the present invention provides a method of activating a smart card comprising; upon receiving power from a terminal, beginning a start-up routine, and as part of the start-up routine before beginning an input/output routine by which an application stored on the smart card receives a command from the terminal, determining whether an uncompleted transaction remains pending in the smart card. The presence of an uncompleted transaction pending in the smart card may be determined by interrogating a transaction record indicator stored in non-volatile memory. Upon determining that an uncompleted transaction remains pending in the smart card, the method further comprises; locating a transaction data record in memory, determining from the transaction data record an original data object which was the subject of the uncompleted transaction, and recreating the original data object from the transaction data record. After recreating the original data object, the method may include resetting the transaction record indicator, deleting the transaction data record, and completing the start-up routine.

In yet another related aspect of the present invention, a method is provided in which upon completing the start-up routine, an answer-to-reset (ATR) signal is sent from the smart card to the terminal. If the ATR signal includes multiple bytes, upon receiving power from a terminal and beginning the start-up routine, the smart card may send only a part of the ATR signal to the terminal before determining whether an uncompleted transaction remains pending in the smart card.

In still another aspect, the present invention provides a method for activating a smart card to accomplish a transaction with a terminal, the smart card comprising an non-volatile memory containing a file directory storing one or more data records and containing a transaction record indicator, the method comprising; upon receiving power in the smart card, interrogating the transaction record indicator, and upon determining in response to the interrogation of the transaction record indicator that a transaction record in stored in memory, interrogating the file directory to locate the transaction record. The data record(s) stored in the file directory may include a type field indicating whether or not a data record is a transaction data record.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 illustrates an exemplary smart card;

Fig. 2 illustrates the integrated circuit portion of the exemplary smart card in some additional detail;

Fig. 3A and 3B conceptually illustrate a conventional stored value transaction on a smart card;

Fig. 4 conceptually illustrates a smart card memory portion having data records and data objects adapted for use within the present invention;

Fig. 5 conceptually illustrates a data record structure adapted for use with the present invention;

Fig. 6 is a method flowchart generally illustrating the one aspect of the present invention;

Fig. 7 conceptually illustrates a smart card memory portion having data records and data objects accessed and manipulated by the method illustrated by the flowchart in Fig. 6;

Fig. 8 is a method flowchart generally illustrating another aspect of the present invention; and

Fig. 9 is a method flowchart generally illustrating yet another aspect of the present invention.

DETAILED DESCRIPTION

Smart cards hold great potential. To date, much of this potential is unrealized. One reason for this underachievement, is the absence of adequate mechanisms assuring data integrity. As explained above, the interruption of a transaction, whether purposefully caused or occurring accidentally, can result in a loss of data integrity. The term "transaction" as used herein broadly describes any exchange of data between the smart card and a terminal, or between one application on the smart card and the OS or another application.

The present invention provides a mechanism by which the state of the smart card memory is rolled-back to a point where data integrity is assured following an interrupted transaction. For ease of reference, the program code implementing this mechanism is called a "transaction manager."

The transaction manager may be located in one or more applications, but as presently preferred, the code capable of implementing the transaction manager is resident in the smart card operating system (OS). The OS is preferably stored in ROM, but may be stored, wholly or in part in EEPROM. The OS is a "true" OS in the sense that it does not execute any command received from a terminal. Rather, the OS provides an I/O routine by which commands are transferred from the terminal to an application running on the smart card, and provides a number of functions which may be called by any one of the smart card applications. The term "function," as used in the context of OS capabilities, denotes a set of related operational functions, instructions, processes, and/or definitions which have been conveniently grouped or identified together according to their nature or operational relationships.

The transaction manger is one such OS function and may be called by any application or any other OS function running on the smart card. The term "call" or "calling" is used throughout to broadly describe a relationship between two pieces of code in which one piece invokes the other. The transaction manager may be called anytime a data record stored in memory is to be changed. Use of the transaction manager is permissive in many instances, but may be mandatory in others according to the protocols of the OS or application(s) being run.

The transaction manager, as explained below, protects the integrity of data records

stored on the smart card from transaction failures and transaction related attacks. In effect, the transaction manager ensures that once the smart card commits to a transaction, the card will either successfully create a new data record as a result of the completed transaction, or roll-back the "old" data record in the event that the transaction is interrupted.

In order to better explain the transaction manager, some context is needed. While the context provided below illustrates the nature and operation of the transaction manager, the transaction manager is not limited to this specific context.

Fig. 4 illustrates the EEPROM portion of a smart card memory. EEPROM is the most common form of Read/Write memory used in present portable tokens. However, the present invention contemplates any form of Read/Write memory, including flash memory, in place of the given example. The EEPROM 40 of Fig. 4 comprises a set of global data objects 41. These global data objects may be defined by the smart card manufacturer, the smart card issuer, the smart card operating system, and/or one or more applications running on the smart card. Typically, a portion of EEPROM is reserved or set aside for the storage of global data objects 41. The global data objects may include just about anything from a single bit flag to an entire application.

In one aspect of the present invention, the transaction manager relies on a global data object called a global transaction (TRANS) indicator 43. In its most basic form, the global transaction indicator is a single bit flag indicating whether or not a transaction data record (TRANS REC) is stored in memory. (Alternatively, the transaction indicator might be implemented as a data record in the file directory, provided sufficient security mechanisms are in place to assure the integrity of such a data record).

EEPROM 40 further comprises a file directory 44 containing one or more data records (0, 1, . . . , N-1, N). Each data record indicates a data object stored in memory. Like the global data objects, data objects stored in memory and indicated by a data record in file directory 44 may take just about any form.

The transaction data record of the present invention may be stored in a memory location defined within the global data objects, or in some other set-aside memory location in non-volatile memory. However, in one preferred embodiment of the present invention, the transaction data record is stored in file directory 44. Since each data record in the file directory is stored with a known structure or format, a transaction data record will have the

same basic structure as all other data records. Data records within a file directory may be distinguished, for example, by their type field. While not required as part of the present invention, use of a standard data record structure provides notable benefits.

For example, in order to properly reconstruct a memory management record for the smart card, every relevant file, record, data object, and application must be stored in non-volatile memory in some form recognizable to a routine which constructs the memory management record. Only data stored in non-volatile memory survives a loss of power in the smart card. Accordingly, all data intended to be "persistent" must be stored in non-volatile memory. (For purposes of this explanation, the non-volatile memory element is presumed to be an EEPROM. However, other types of non-volatile memory might be used).

Thus, a data record structure is defined for all data stored in EEPROM which is intended to be persistent. This data record structure is recognized by the OS and by applications, and is specifically recognized by a routine which creates the memory management record. As one of ordinary skill will understand, the exact nature, size, and characteristics of this "standard" data record structure can be left to the individual programmer. The example which follows is merely a presently preferred example.

A data record may be an application, a file, or any other type of persistent data. As conceptually illustrated in Fig. 4, an original data record 45 stored in file directory 44 may reference an actual data object 46 associated with original data record 45, but stored elsewhere in memory. Fig. 5 illustrates a 16-byte data record structure comprising a 2-byte ID field, a 1-byte ownership field, a 1-byte type field, a 4-byte data field, a 2-byte data length field, and a 6-byte label field. As presently preferred the type field and the label field are user definable. That is, an application's programmer may use these data record fields for any purpose whatsoever. The memory manager, and the OS in general, do not care what these fields contain. They are merely variable data fields associated with a data record. As examples, the type field might indicate whether the data record is an application, a file, or some other type data object, like a transaction data record. The label field might indicate the access type for the data record.

The ID field identifies the data record within the file system administered by the OS. The ownership field includes ownership information. In the present example, the

ownership field of the data record contains the unique ownership byte previously described. Only the OS may access and define the ID and ownership fields in each data record.

The data field and the data length field are related within each data record. The data length field specifies the size of the data field. In the preferred embodiment, the data field is allocated 4 bytes, it's maximum data size. Thus, if the data length field indicates that the data is 4 bytes or less in size, then the data field stores the actual data associated with the data record. If, however, the data length field indicates that the data field is greater than 4 bytes in size, the data field stores a 4-byte data pointer indicating the beginning address, elsewhere in EEPROM, at which the actual data may be found, for example, the actual data 46 shown in Fig. 4.

Some transactions will not require the services of a transaction manager. These transactions do not create or change a significant data record in the smart card, and as such do not require the security offered by use of the transaction manager.

However, the method illustrated by Fig 6 assumes a transaction 60 in which a significant data object is changed from one value to another, as in the case where the smart card is being used as a stored value card. Thus, at some point, the application performing this transaction will call the transaction manager 61.

Once called, the transaction manager creates a transaction data record 71 in file directory 44. Assuming use of data records having the form discussed above, and with the example shown in Fig 4, transaction data record 71 references a copy of original data object 45. The copy is denoted by 45'. Depending on the size of original data object 45, the process of copying it has one of two effects. First, if the actual data associated with original data object 45 is less than 4 bytes, then transaction record 71 merely references a copy 45' of the original data object. See, 71A in Fig. 7. If, however, original data object 45, has associated data greater than 4 bytes in size, i.e., original data record 45 references actual data object 46, then transaction data record 71 references a copy of the original data record 45' and a copy of the actual data 46' associated with the original data record 45. See, 71B in Fig. 7.

Returning to the flowchart in Fig. 6, once the original data record, regardless of data size or the exact copying process, has been successfully copied into the transaction

record 62, the transaction manager sets a global transaction indicator in memory 63. This indicator may be located anywhere in non-volatile memory, but as presently preferred is consistently stored at a memory location found in the space storing global data objects. Once set, the global transaction indicator indicates that a transaction is pending and that a transaction record associated with the, as yet uncompleted transaction, is stored in memory.

At this point, it must be noted that the present invention is well adapted to securely administer a sequence of nested transactions. That is, the transaction manger may be called and effectively used to securely handle a second and subsequent transactions within the first and subsequent transactions. In order to accomplish this, the transaction manager is repeatedly called and a series of transaction records are created in the file directory. Assuming the smart card loses power before the plurality of transactions are completed, the smart card start-up merely resolves each transaction record as it scrolls through the file directory, as explained in greater detail below.

Alternatively, the global transaction indicator may prioritize a nested sequence of transactions. Thus, upon re-powering the smart any unresolved transactions are cleared according to their creation priority, for example, last-created, first-cleared.

Returning to the single transaction described with reference to Figs. 6 and 7, once the global transaction indicator is set, the application returns and continues the transaction 64. Upon successfully completing the transaction 65, the application again calls the transaction manager 66, which deletes the transaction record 67, and resets the global transaction indicator 68.

By this process, the smart card is always aware of the presence of an uncompleted transaction. Accordingly, the smart may will not operate upon re-powering the smart card in a subsequent session with a terminal until all pending transaction have been cleared. In this manner, all "old" data is rolled back before a subsequent transaction session is begun. Data integrity is thus assured.

This ability is further explained below with reference to the flowchart shown in Fig. 8. In the method illustrated by this flowchart, it is assumed that a smart card has just been activated within a terminal. As part of a start-up routine, the smart card OS determines whether an unresolved transaction is present 81. If an unresolved transaction is

detected, the transaction record is located 82, the original data object, in whatever state it is presently found, is deleted 83, and then recreated using the transaction record 84. The term "original" is used here to indicate the data object being modified when the transaction was interrupted.

Once the original data object has been recreated, the global transaction indicator is reset 85, and the transaction record is cleared 86. The smart card is now able to return to the start-up routine having "rolled-back" correct data into the original data object.

Continuing the example described in Figs. 4 and 7, the roll-back routine of Fig 8 will be further described in the context of a stored value application. In this example, the original data object 45, along with its associated data 46, contain stored value information. It is further assumed that the smart card was yanked during a previous stored value transaction in an attempt to create an undefined, or wrongly defined stored value in memory.

Thus, in the next subsequent activation, the smart card enters its start-up routine. During the start-up routine, the OS looks at the global transaction indicator 43. Since this flag is set, the OS knows a transaction remains pending from the previous terminal session. In this case, the stored value data, i.e. the original data object 45/46 remains incorrectly defined or undefined.

In order to maintain data integrity, the smart card rolls-back the "old" stored value information found in the transaction record. Accordingly, transaction record 71 is located by, for example, having the OS scroll through the file directory looking for any data record having a type field indicator corresponding to a transaction record. Transaction record 71 indicates the location in memory storing a copy of original data object 71B. The original data object record stored in the file directory, with its unknown data state, is deleted and replaced (or rewritten) by the contents of transaction data record 71. After rolling-back the original data object, the global transaction indicator 43 is cleared, and transaction record 71, with its associated data object 71B, is deleted.

Another aspect of the present invention is described with reference to the flowchart in Fig. 9. As mentioned above, ISO-7816 defines the ATR signal used by conventional smart cards to initiate communication with a terminal. Given the relatively simple nature of conventional smart card operation, the existing ATR protocol does not necessarily

contemplate a lengthy "start-up" routine including rolling back data for one or more uncompleted transactions. It is possible, therefore, for the terminal to prematurely terminate a session when an ATR signal is not received within an anticipated period of time.

The conventional ATR signal comprises multiple bytes of data. The present invention uses this fact, to avoid "timing-out" during the Reset/ATR exchange between a terminal and a smart card. For example, looking a Fig. 9, upon receiving power and a reset signal, the smart card begins a start-up routine 91. Soon after beginning the start-up routine, the smart card returns one or more bytes of the ATR signal 92 to essentially "hold" the terminal while completing the start-up routine 93. Once the start-up routine is complete, the smart card sends the remaining portion of the ATR signal 94 before entering an I/O loop, or performing some other function.

The transaction manager of the present invention precludes successful card yank attacks, and generally ensures data integrity otherwise threatened by transaction interruptions. This feature is particularly important in smart cards running multiple applications. Since each application may read/write data into memory, and potentially read/write data common to other applications and the smart card OS, data integrity must be assured at all times, uncompleted transactions notwithstanding.

The present invention has been described using several examples. The context needed to explain the transaction manager includes some assumptions regarding data record use and storage, data record structure, data object organization in memory, and the type of memory (EEPROM) commonly used. These contextual aspects, among others, are used to explain the broader concepts associated with the use of a transaction manager within a smart card. The transaction manager is limited to the disclosed examples and teaching context, but is defined by the attached claims.

What is claimed is:

1. A method of securely performing a transaction on a smart card receiving a command from a terminal, wherein the command requires changing an original data object stored in smart card memory, the method comprising:

copying the original data object into a transaction record; and

setting a transaction record indicator.

2. The method of claim 1, further comprising:

after setting the transaction record indicator, changing the original data object; and

after successfully changing the original data object, deleting the transaction record and resetting the transaction record indicator.

3. The method of claim 2, wherein the smart card memory comprises an EEPROM.

4. The method of claim 3, wherein the transaction record indicator comprises a global data object established in a reserve portion of EEPROM.

5. The method of claim 3, wherein the original data object comprises an original data record.

6. The method of claim 5, wherein the original data record is stored in a file directory established in EEPROM.

7. The method of claim 6, wherein copying the original data object into the transaction record comprises creating a transaction data record in the file directory.

8. The method of claim 7, wherein the transaction data record identifies an address in memory at which a copy of the original data record is located.

9. The method of claim 3, wherein the original data object comprises an original data record stored in a file directory in memory, and additional data associated with the original data record.

10. The method of claim 9, wherein copying the original data object into the transaction record comprises creating a transaction data record in the file directory, wherein the transaction data record contains a field identifying at least one address in memory at which a copy of the original data record and a copy of the additional associated data are located.

11. A method of executing a transaction between a terminal and a smart card, the

smart card comprising a microprocessor and a memory, the memory storing an application and an operating system (OS), the OS comprising a transaction manager, and the method comprising:

receiving a command in the OS from the terminal, wherein the command requires changing an original data object;

communicating the command from the OS to the application;

calling the transaction manager from the application;

by operation of the transaction manager, copying the original data object into a transaction record; and

setting a transaction record indicator in memory.

12. The method of claim 11, further comprising

after setting the transaction record indicator, returning to the application and changing the original data object in accordance with the command;

after successfully changing the original data object, calling the transaction manager; and

by operation of the transaction manager, deleting the transaction record and resetting the transaction record indicator.

13. The method of claim 11, wherein the command comprises a series of commands received in the OS from the terminal and communicated to the application, wherein during the series of commands the application calls the transaction manager.

14. The method of claim 13, wherein the OS is incapable of executing any one of the series of commands received from the terminal.

15. The method of claim 11, wherein the memory contains a file directory storing persistent data records, and wherein copying the original data object into the transaction record comprises creating a transaction data record and storing the transaction data record in the file directory.

16. A method of activating a smart card, comprising:

upon receiving power from a terminal, beginning a start-up routine;

as part of the start-up routine and before beginning an input/output routine by which an application stored on the smart card receives a command from the terminal, determining whether an uncompleted transaction remains pending in the smart card.

17. The method of claim 16, wherein determining whether an uncompleted transaction remains pending in the smart card comprises interrogating a transaction record indicator stored in non-volatile memory.

18. The method of claim 17, whereupon determining that an uncompleted transaction remains pending in the smart card, the method further comprises:

locating a transaction data record in memory;

determining from the transaction data record an original data object which was the subject of the uncompleted transaction; and

recreating the original data object from the transaction data record.

19. The method of claim 18, further comprising:

after recreating the original data object, resetting the transaction record indicator, deleting the transaction data record, and completing the start-up routine.

20. The method of claim 19, wherein completing the start-up routine comprises sending an answer-to-reset (ATR) signal from the smart card to the terminal.

21. The method of claim 16, wherein the start-up routine comprises sending a multiple byte answer-to-reset (ATR) signal from the smart card to the terminal.

22. The method of claim 21, wherein upon receiving power from a terminal and beginning the start-up routine, the smart card sends part of the ATR signal to the terminal before determining whether an uncompleted transaction remains pending in the smart card.

23. The method of claim 18, wherein the smart card comprises a memory storing the application and an operating system (OS), the OS performing the start-up routine and comprising a transaction manger, wherein the method further comprises:

upon determining from the transaction record indicator, as part of the start-up routine that an uncompleted transaction remains pending in the smart card, calling the transaction manager; and

by operation of the transaction manger, locating the transaction data record in memory;

determining from the transaction data record the original data object which was the subject of the uncompleted transaction;

recreating the original data object from the transaction data record;

resetting the transaction record indicator; and

returning to the start-up routine.

24. A method of changing an original data object in a smart card, the smart card comprising a memory storing an application and an operating system (OS), the OS comprising a transaction manager, and the method comprising:

calling the transaction manager from either the OS or the application; and

by operation of the transaction manager, copying the original data object into a transaction record, and setting a transaction record indicator.

25. The method of claim 24, further comprising:

maintaining the transaction record until such time as the original data record has been successfully changed.

26. The method of claim 25, further comprising:

only after successfully changing the data object, deleting the transaction data record and resetting the transaction data record indicator.

27. A method of activating a smart card to accomplish a transaction with a terminal, the smart card comprising an non-volatile memory containing a file directory storing one or more data records and containing a transaction record indicator, the method comprising:

upon receiving power in the smart card, interrogating the transaction record indicator; and

upon determining in response to the interrogation of the transaction record indicator that a transaction record in stored in memory, interrogating the file directory to locate the transaction record.

28. The method of claim 27, wherein each data record stored in the file directory comprises a type field, the type field indicating whether or not a data record is a transaction data record.

29. The method of claim 26, further comprising:

locating an original data record stored in memory having questionable data; and replacing the questionable data with data from the transaction data record.

30. The method of claim 29, further comprising:

following replacement of the questionable data, deleting the transaction record and changing the data state of the transaction record indicator to indicate that a transaction

data record is no longer stored in memory.

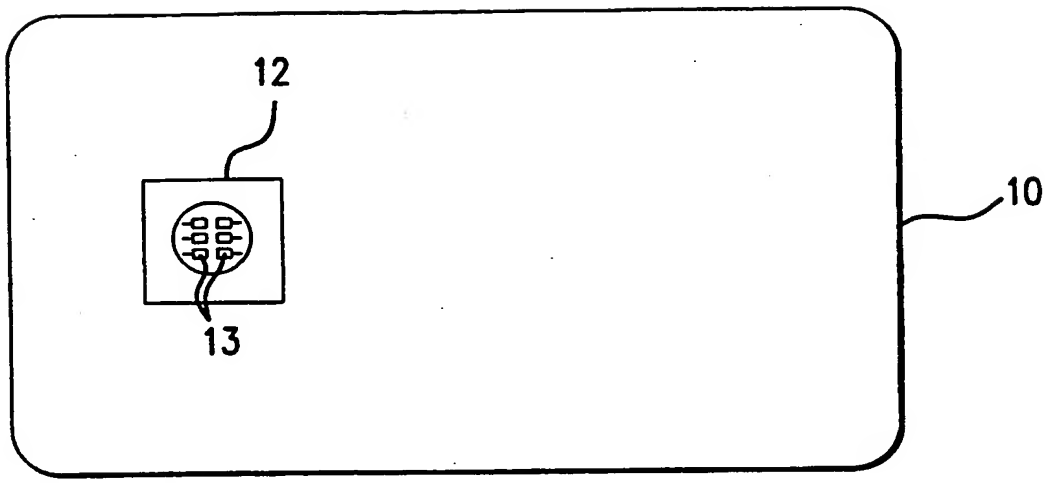

12
10
13

FIG.1

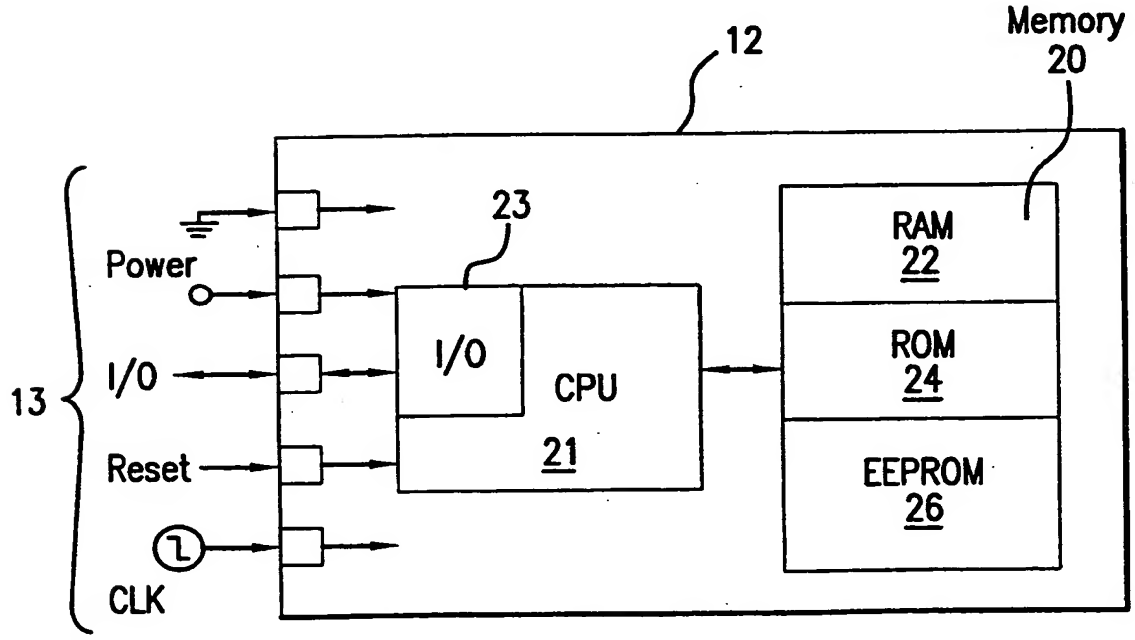

12
Memory
20
23
Power
I/O
13
Reset
CLK
I/O
CPU
21
RAM
22
ROM
24
EEPROM
26

FIG.2

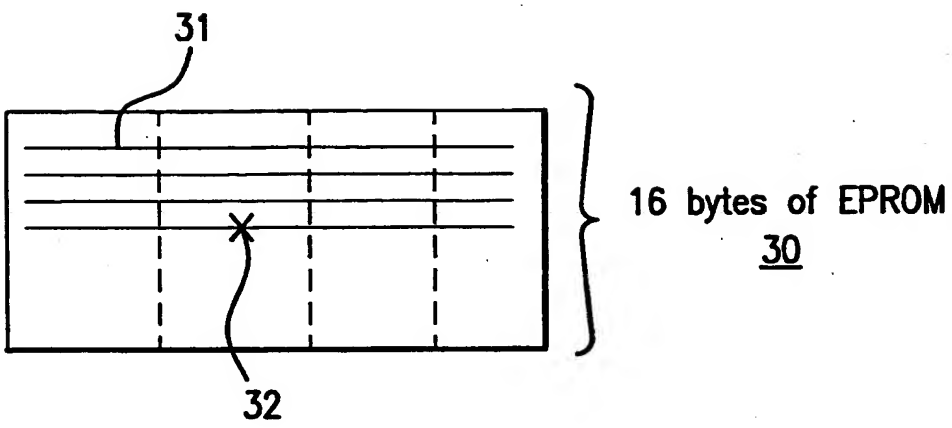


FIG.3A

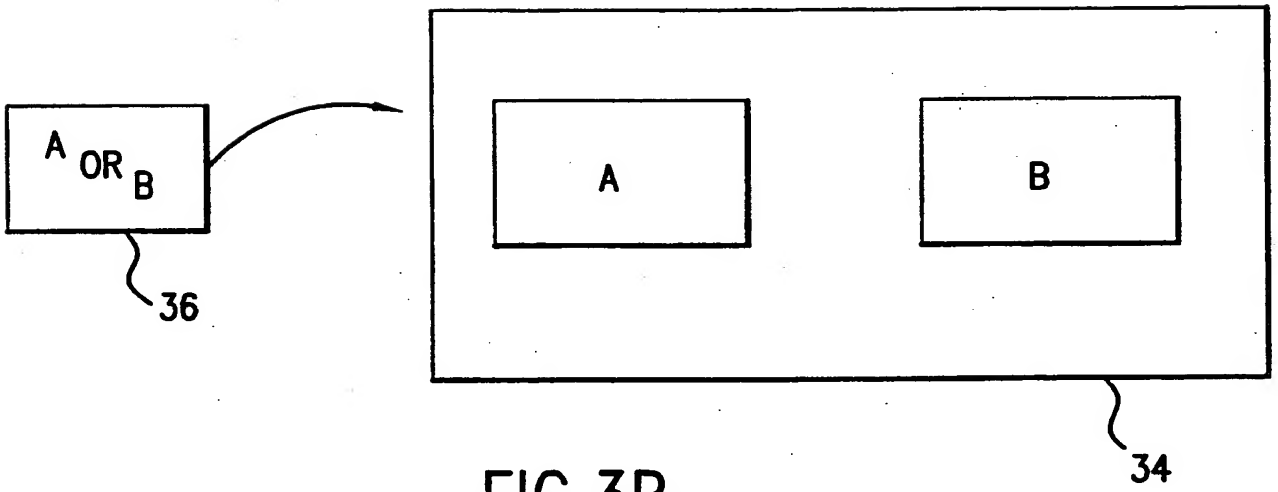


FIG.3B

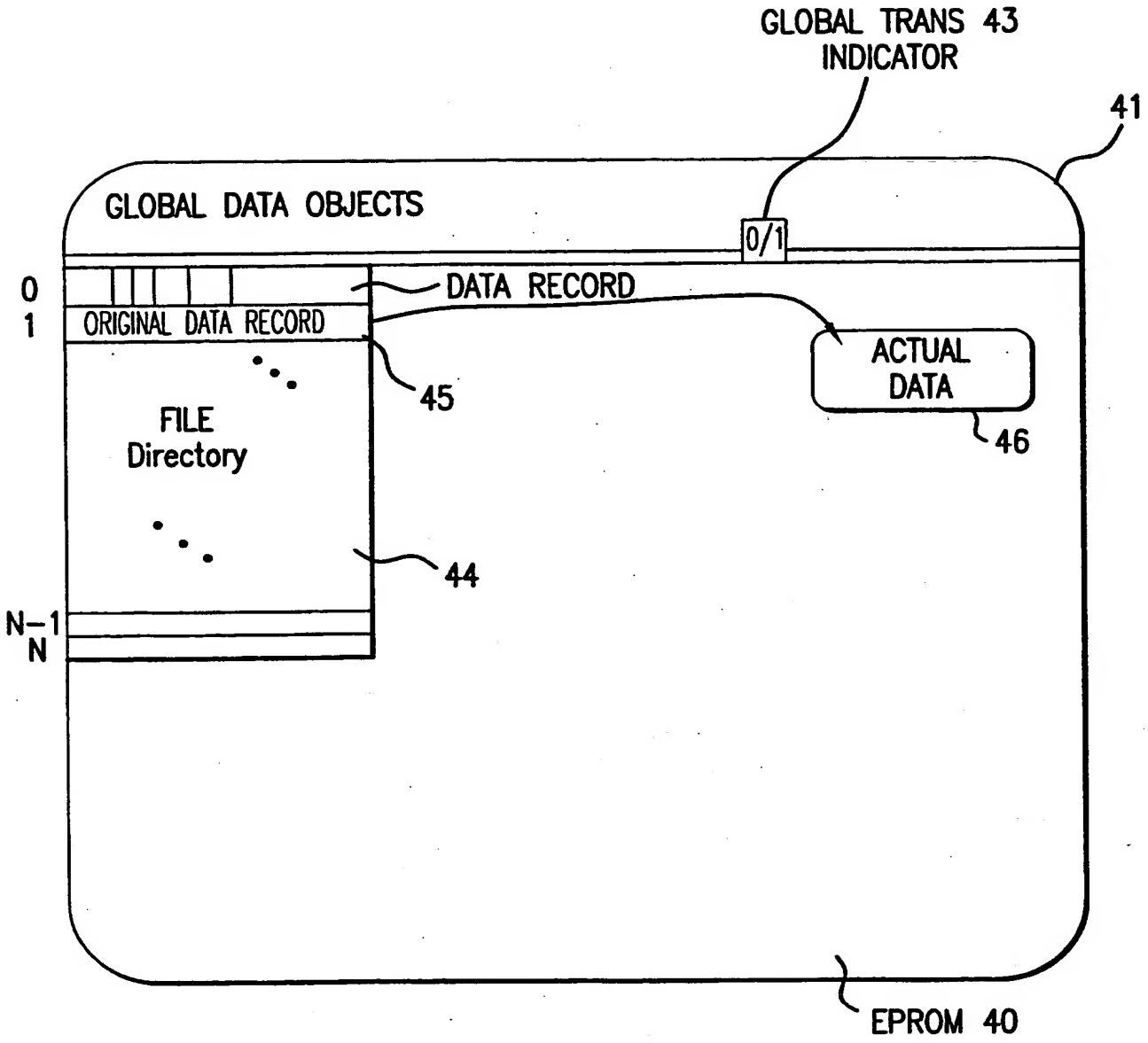

GLOBAL TRANS 43
INDICATOR
41
GLOBAL DATA OBJECTS
0/1
0
1
DATA RECORD
ORIGINAL DATA RECORD
45
ACTUAL
DATA
46
FILE
Directory
44
N-1
N
EPROM 40

FIG.4

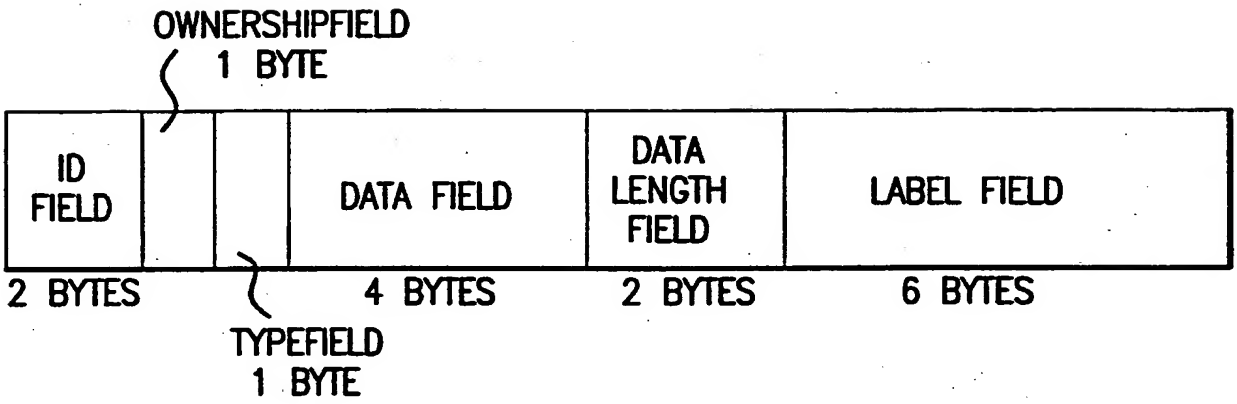

OWNERSHIPFIELD
1 BYTE
ID FIELD
DATA FIELD
DATA LENGTH FIELD
LABEL FIELD
2 BYTES
4 BYTES
2 BYTES
6 BYTES
TYPEFIELD
1 BYTE

FIG.5

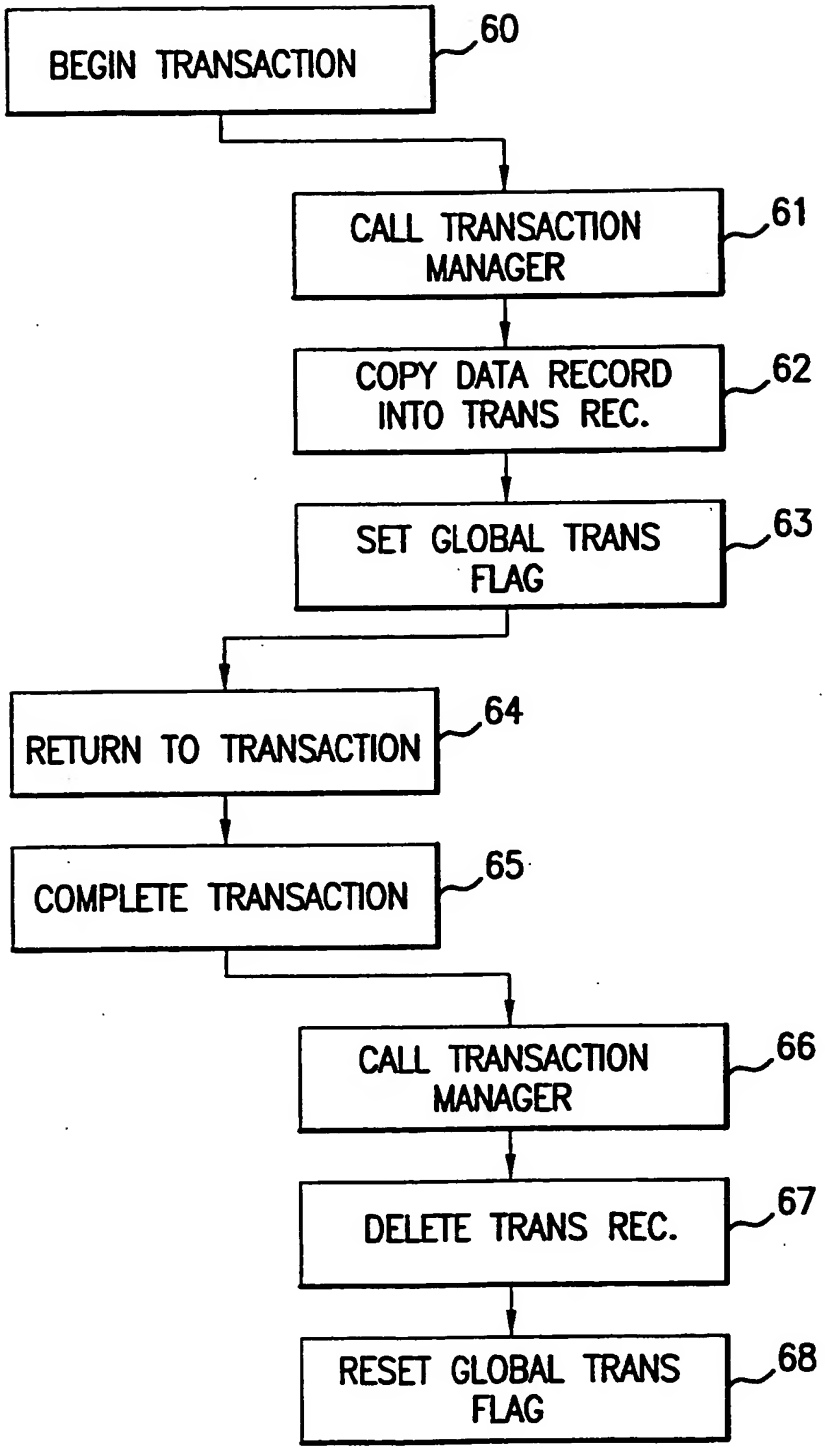


FIG.6

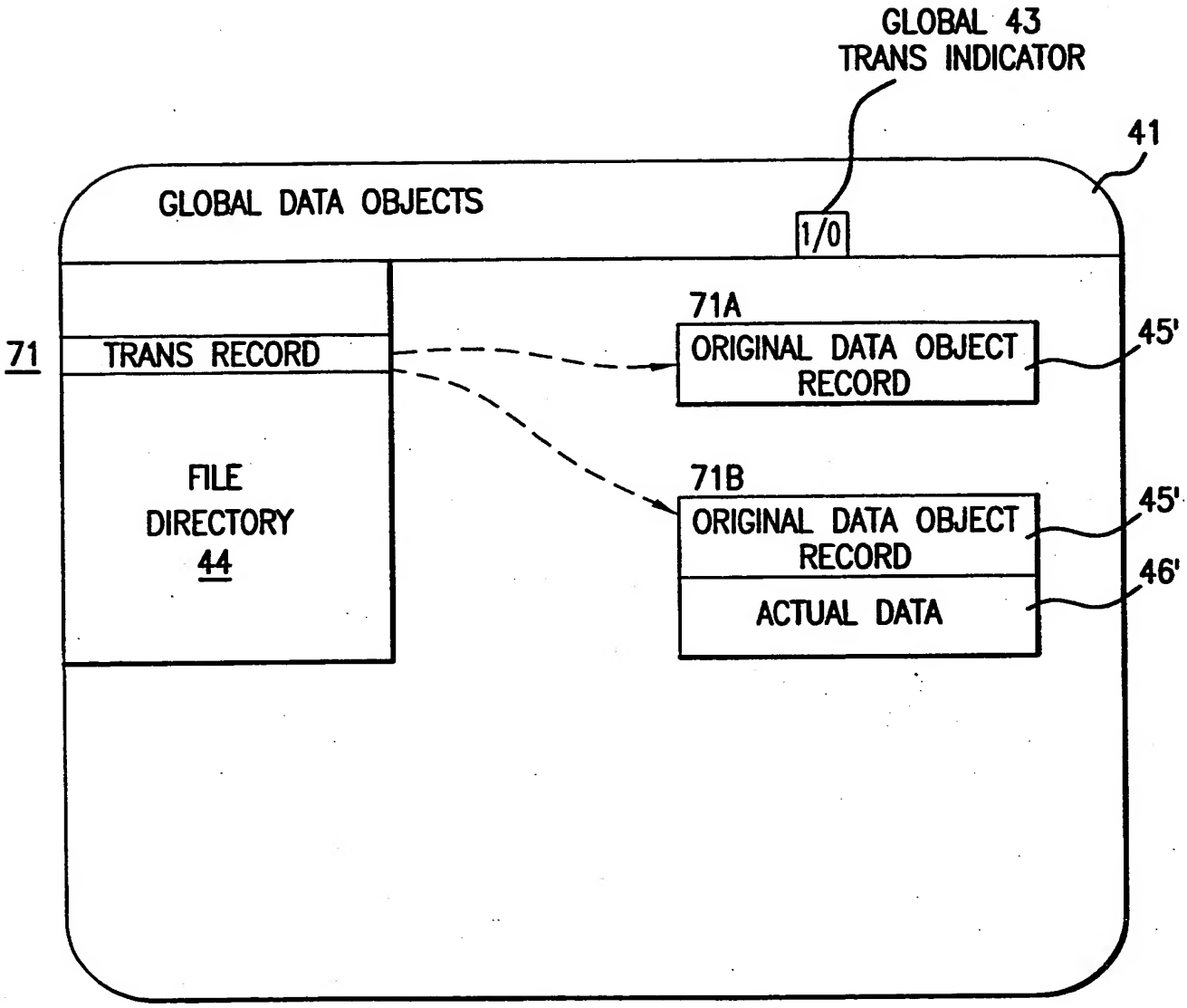


FIG.7

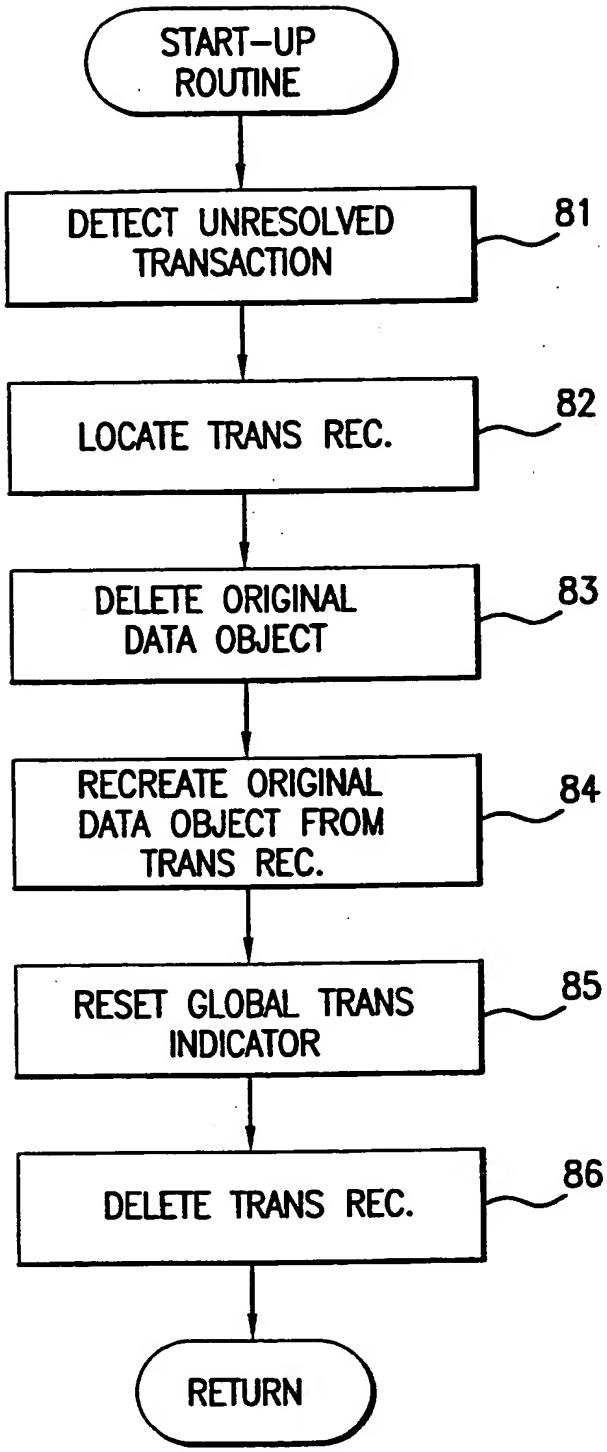

START-UP ROUTINE
DETECT UNRESOLVED TRANSACTION
81
LOCATE TRANS REC.
82
DELETE ORIGINAL DATA OBJECT
83
RECREATE ORIGINAL DATA OBJECT FROM TRANS REC.
84
RESET GLOBAL TRANS INDICATOR
85
DELETE TRANS REC.
86
RETURN

FIG.8

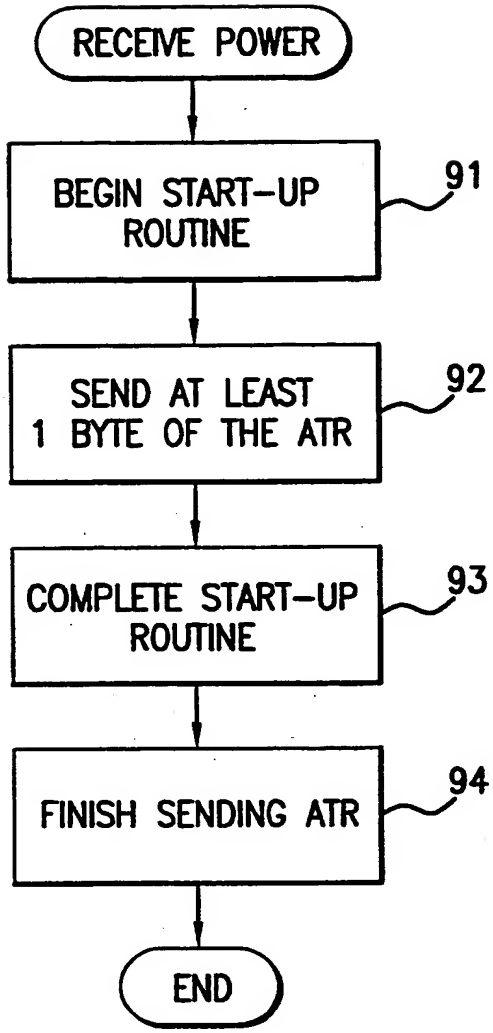

RECEIVE POWER
BEGIN START-UP ROUTINE
91
SEND AT LEAST 1 BYTE OF THE ATR
92
COMPLETE START-UP ROUTINE
93
FINISH SENDING ATR
94
END

FIG.9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/00084

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06K 19/06, 05/00, 07/00
US CL :235/380, 382, 375, 486, 492, 487
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 235/380, 382, 375, 486, 492, 487

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

None

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

USPTO APS EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,930,363 A (Standford et al.) 27 July 1999 (27-07-1999), the entire reference.	1-30
Y	US 4,053,735 A (Foudos) 11 October 1977 (11-10-1977), the entire reference.	1-30

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 09 APRIL 2000	Date of mailing of the international search report 21 APR 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer THIEN LE Telephone No. (703) 305-3500

Form PCT/ISA/210 (second sheet) (July 1998)★